Information and Software Technology 99 (2018) 93–97

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Not all bug reopens are negative: A case study on eclipse bug reports

Qing Mi*[a], Jacky Keung[a], Yuqi Huo[b], Solomon Mensah[a]

[a] Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong
[b] School of Information, Renmin University of China, Beijing, China

ARTICLE INFO

Keywords:
Non-negative bug reopen
Bug report
Reopen cycle
Data quality
Open source software
Empirical software engineering

ABSTRACT

Context: We observed a special type of bug reopen that has no direct impact on the user experience or the normal operation of the system being developed. We refer to these as *non-negative bug reopens*.
Objective: Non-negative bug reopens are novel and somewhat contradictory to popular conceptions. Therefore, we thoroughly explored these phenomena in this study.
Method: We begin with a novel approach that preliminarily characterizes non-negative bug reopens. Based on bug reports extracted from Eclipse Bugzilla, we then examined a case study to compare non-negative and regular bug reopens using the Wilcoxon-Mann-Whitney test.
Results: The results show that non-negative bug reopens are statistically significantly different than regular bug reopens, based on their survival times and the number of developers involved in the entire debugging process.
Conclusion: Taking into account the significant differences, we suggest that the effects of non-negative bug reopens should be considered in future research in related areas, such as bug triage and reopened bug prediction.

1. Introduction

A software bug that has been REOPENED for further processing after being RESOLVED/VERIFIED/CLOSED is known as a*reopened bug*. Approximately 6%–10% of total bugs are eventually reopened [1], but the problem of bug reopening has only recently attracted the attention of the software engineering community. Zimmermann et al. [2] characterized the reopening process using a mixed approach. Shihab et al. [3] constructed decision trees to predict whether a bug will be reopened. Xia et al. [4] used a combination of three classifiers to improve the performance of reopened bug prediction.

Research on bug reopens is important to the software engineering community as it provides capabilities needed to: 1) Plan maintenance efforts taking into account the reopening rate [1]; 2) Characterize the actual quality of the bug fixing process [2]; 3) Prepare practitioners to think twice before closing a bug [3,4]. Our work here is complementary to previous studies.

Bug reopens are normally considered to be negative since they take longer to be resolved [1,3,4], cause rework for already-busy developers [1,4], and degrade the user-perceived quality of the software product [3]. However, the results of this study indicate otherwise. Table 1 lists a variety of reopen reasons determined in our previous investigation [1]. Further analysis revealed that a significant proportion of bug reopens have no direct effect on the user experience or the normal operation of the system being developed. For instance, R4, R6, and R7 in Table 1 may be considered to be either *non-essential* or *non-negative* bug reopens. A case study on Eclipse bug reports is examined; we show, statistically, that non-negative bug reopens are significantly different than regular bug reopens, based on their survival times and the number of developers involved in the entire debugging process. Thus, we propose that non-negative bug reopens should not be treated as equivalent to regular bug reopens.

2. Identifying non-negative bug reopens

We begin by presenting an overview of the reopen cycle and its corresponding taxonomy. Three patterns are then proposed to characterize non-negative bug reopens.

2.1. Reopen cycle and its representation

Our research is based on Bugzilla,[1] a popular system used to track reported software bugs [5]. Fig. 1 shows the typical timeline for a Bugzilla bug. Note that the lifetime for a regular bug (the upper arrow in Fig. 1) is considered to be completed when it first reaches the status RESOLVED/VERIFIED/CLOSED, whereas a reopened bug (the lower arrow in Fig. 1) survives for at least one more reopen cycle as depicted in the dotted rectangles in Fig. 1.

* Corresponding author.
E-mail addresses: Qing.Mi@my.cityu.edu.hk (Q. Mi), Jacky.Keung@cityu.edu.hk (J. Keung), bnhony@ruc.edu.cn (Y. Huo), smensah2-c@my.cityu.edu.hk (S. Mensah).
[1] https://www.bugzilla.org.

https://doi.org/10.1016/j.infsof.2018.03.006
Received 4 October 2016; Received in revised form 29 January 2018; Accepted 5 March 2018
Available online 06 March 2018
0950-5849/

Table 1
Root causes for bug reopens. The reason IDs are in accordance with those used in our previous work [1]. The eighth category of reopen reasons, rare causes, is intentionally omitted from this study.

Classification	ID	Reopen reason	Description	Type
Unsuccessful fix	R1	Bug reporter's unclear description	The bug-related information (e.g., steps to reproduce) provided by the reporter may be incomplete, ambiguous, or inaccurate.	T2
	R2	Developer's negligence	The assigned developers do not treat the reported bug seriously.	T2
	R3	Introduction of additional problems	The bug has been resolved, yet some unexpected problems (e.g., instability) are introduced into the system.	T1
	R5	Incomplete fix	The bug is thought to be resolved, yet someone reproduces the issue on a later version.	T1
Further improvements and other reasons	R4	A better solution	The bug has been resolved, yet a better solution is found afterwards (typically by the same developer).	T1
	R6	Misapprehension	The bug has been resolved, yet the reporter mistakenly considers that the issue still exists.	T1, T4
	R7	Inappropriate status	The bug has been resolved, yet the accompanying attributes (e.g., resolution) may be improperly or incorrectly completed.	T2, T3, T4

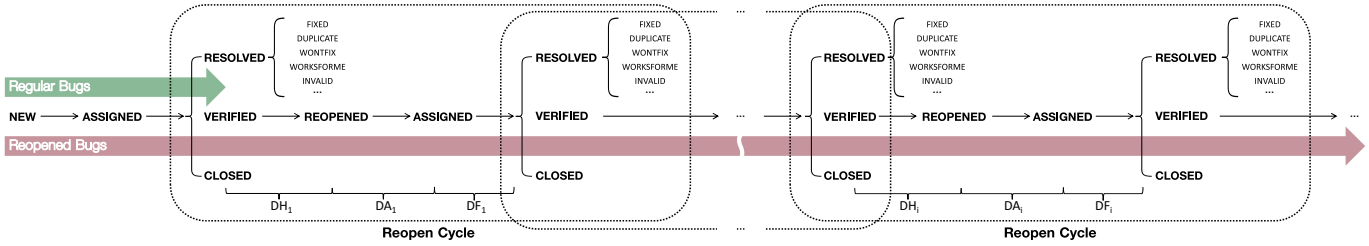

Fig. 1. Typical timeline for a Bugzilla bug.

A typical reopen cycle consists of four stages: RESOLVED/VERIFIED/CLOSED, REOPENED, ASSIGNED, and then back to RESOLVED/VERIFIED/CLOSED. Every time a bug goes through the RESOLVED/VERIFIED/CLOSED stage, one of the possible resolutions (FIXED, DUPLICATE, WONTFIX, WORKSFORME, INVALID, NOT_ECLIPSE, LATER, and REMIND) is used to describe what happened to the bug. Ideally, all resolutions should be properly verified, yet many bugs end up with the status RESOLVED in practice.

To represent the reopen cycle, we followed the method detailed by Jongyindee et al. [6], which is designed to symbolize bug history in the form of status changes (e.g., NEW ⇒ ASSIGNED ⇒ RESOLVED (FIXED)). Because the objective of our study differed from that of Jongyindee et al., we used a relatively concise format that ignored bug status and was restricted only to resolutions. Accordingly, each reopen cycle can be characterized as a pair of resolutions, for instance, WORKSFORME ⇒ FIXED. In this scheme, "⇒" stands for sequence rather than causality.

2.2. Taxonomy of reopen cycles

Given that reopen cycles served as the basis for this study, a classification framework was developed to facilitate further analysis.

Based on whether a check-in to a code repository is involved, resolutions fall broadly into two groups: Fixed (FIXED) and Non-Fixed (DUPLICATE, WONTFIX, WORKSFORME, INVALID, NOT_ECLIPSE, LATER, and REMIND). Accordingly, all reopen cycles can be divided into the following four types: **T1.** Fixed ⇒ Fixed; **T2.** Non-Fixed ⇒ Fixed; **T3.** Fixed ⇒ Non-Fixed; **T4.** Non-Fixed ⇒ Non-Fixed.

As shown in Table 1, R1 and R2 are the typical scenarios for T2, whereas R3, R4, and R5 fall into T1. However, R6 and R7 cannot be classified into any single type; these were the main targets for our identification as non-negative bug reopens.

2.3. Patterns of non-negative bug reopens

The primary criteria for non-negative bug reopens are that neither the user experience nor the system execution performance is significantly affected. Therefore, we focused solely on reopen cycles without code modification; thus, only T3 and T4 were within the scope of our research. Accordingly, a total of three patterns were proposed to identify non-negative bug reopens:

- **P1.** Fixed ⇒ INVALID/WORKSFORME/NOT_ECLIPSE (T3)
  The pattern is proposed to partially identify R7. In some circumstances, reported bugs are erroneously labeled as FIXED when no code churns have actually been made. The assigned developer realizes his or her oversight and reopens the bug to provide a proper resolution.
- **P2.** Non-Fixed ⇒ The same resolution (T4)
  The pattern is comprised of a pair of consistent resolutions (e.g., INVALID ⇒ INVALID) and is used to partially identify: 1) R6 (where the bug is closed with the same resolution after being mistakenly reopened) and 2) R7 (where the bug maintains the same resolution after being reopened to correct the accompanying attributes (e.g., severity and priority)).
- **P3.** INVALID/WORKSFORME/NOT_ECLIPSE ⇒ INVALID/WORKSFORME/NOT_ECLIPSE (T4)
  The pattern is used to partially identify R7. In practice, some bug reopens result from developers' incomplete understanding of statuses such as INVALID, WORKSFORME, and NOT_ECLIPSE, that all represent false positives to varying degrees.

3. Case study

We first present a brief introduction to the studied systems and dataset. After that, we describe the case study on Eclipse bug reports

that was carried out to compare non-negative and regular bug reopens.

3.1. Studied systems and dataset

Eclipse[2] is an open source set of tools, projects, and collaborative working groups that provides IDEs and platforms for nearly every language and architecture. It has more than 200 sub-projects, from which we selected the four most active as our research objects: CDT, JDT, PDE, and Platform. The selected projects have been widely investigated in many recent studies and are capable of providing sufficient bug reports for our analysis.

The *Defect Tracking Dataset* of Lamkanfi et al. [7] was used because it contains fine-grained update information regarding the most relevant attributes from Eclipse bug reports,[3] and also because the current study of non-negative bug reopens is based on our latest research (mainly the results from Table 1) [1]. We intentionally retained the use of the same dataset as in the previous work [1].

The dataset contains the bugs that have been reported from January 2006 to March 2011, a statistical summary is shown in Table 2. It can be observed that non-negative bug reopens account for approximately 12.94% of the reopen cycles in the dataset ($\frac{514}{3972}$). Additionally, the average number of non-negative bug reopens was found to be 99.48 per year ($\frac{514}{5.167}$), which is perceived to be moderately significant and worthy of further exploration.

3.2. Feasibility analysis

In this section, we describe the manual examination conducted to preliminarily verify the feasibility and validity of the patterns proposed in Section 2.3.

To examine the non-negative bug reopens presented in Table 2, we proportionally selected 5% from each pattern as target samples based on stratified random sampling. In addition, we gathered and analyzed developer discussions stored in Eclipse Bugzilla to accurately determine the reasons for bug reopens. Note that each case was assessed by two examiners. If they had conflicting opinions on the reason for a given bug reopen, a third examiner was introduced to make the final decision.

Table 3 provides an overview of the assessment. The rows (P1–P3) represent the pattern of the bug reopen. The columns (R1–R7) represent the actual reason for the bug reopen. The standard precision function ($Precision = \frac{TP}{TP+FP}$) was used to evaluate the proposed patterns. Overall, 92% ($\frac{23}{25}$) of bug reopens were correctly classified.

3.3. Comparative analysis

3.3.1. Methodology

To measure how and to what extent non-negative bug reopens differ from regular ones, we used a similar approach to that of Tian et al. [9].

First, 12 features along 4 dimensions (i.e., work habit, bug report, bug fix, and people) were chosen from previous studies [1,3], as shown in Table 4. Next, simple random sampling was used without replacement to select regular bug reopens to achieve the same size as non-negative ones. After that, the Wilcoxon-Mann-Whitney test (significance level $\alpha = 0.01$) was performed to examine whether non-negative bug reopens were statistically significantly different from regular bug reopens based on these features. To determine the magnitude of the differences between the two comparison groups, we adopted the Cliff's Delta statistic as recommended by Kitchenham et al. [10].

3.3.2. Results

The results of the Wilcoxon-Mann-Whitney test (*p*-value column) and the effect size values (*d-value* column) for each feature are shown in Table 4. Our interpretation of the Cliff's Delta value is in line with those of Tian et al. [9], which is considered negligible for $|d\text{-}value| < 0.147$, small for $|d\text{-}value| < 0.330$, medium for $|d\text{-}value| < 0.474$, and large for otherwise.

A significant difference with at least a small effect size (i.e., $p\text{-}value < 0.01$ and $|d\text{-}value| > 0.147$) is found in 2 of the 4 dimensions studied (highlighted in bold in Table 4). The results show that non-negative bug reopens are statistically significantly different from regular ones in the aspects including *duration of survival* (in CDT, JDT, and Platform), *duration of hidden* (in CDT, JDT, and Platform), and *number of developers involved* (in all four projects examined). The boxplots in Fig. 2 are used to visualize the differences between non-negative and regular bug reopens in means and distributions. Generally, non-negative bug reopens involve fewer developers and have shorter life expectancies, which is consistent with the definition of *non-essential* or*non-negative*.

Table 2
Statistical summary of the dataset used in this study.

Project	Total # of bug reports	Reopen cycle				Non-negative bug reopen		
		T1	T2	T3	T4	P1	P2	P3
CDT	5640	318	50	7	54	1	24	5
JDT	10,814	447	257	63	463	37	135	29
PDE	5655	252	54	9	67	4	21	8
Platform	24,775	905	361	71	594	38	169	43
Total	46,884	1922	722	150	1178	80	349	85

Table 3
Results of manual examination. The wrongly identified cases are marked with asterisks.

Non-negative bug reopen	R1	R2	R3	R4	R5	R6	R7
P1	0	0	0	0	0	0	4
P2	1*	0	0	0	0	15	1
P3	0	0	0	0	0	1*	3

4. Discussion

This work is among the first to determine important patterns and underlying features of non-negative bug reopens. We show, statistically, that non-negative bug reopens are significantly different than regular bug reopens. However, no prior work to our knowledge has explicitly considered the differences. The critical caveat is that previous studies may have been biased, directly or indirectly, by the existence of non-negative bug reopens. For instance, we may have overestimated the likelihood of bug reopening, the extent to which bug reopens can negatively impact end users, or even the importance of the area as a whole. In addition, our findings indicate, to some extent, that data extracted from bug tracking systems (the most frequently used resources for bug-related research) may not be sufficiently reliable, which needs to be validated before any analysis is attempted.

Given the above consequences, we encourage researchers engaged in relevant fields (such as reopened bug prediction and effort estimation) to consider the effects of non-negative bug reopens in their future work. Additionally, the very existence of non-negative bug reopens, according to their sample causes (R4, R6, and R7 in Table 1), can be applied as a low-cost heuristic to identify problematic procedures in bug tracking systems (e.g., the lack of clear guidelines and the misuse of the status REOPENED). Generally, the more non-negative bug reopens can be found, the greater the need for better tools to improve the workflow of reported software bugs, while the identification of non-negative bug reopens can be achieved simply by using the method proposed in this study (Section 2.3).

[2] https://www.eclipse.org.

[3] Note that feature requests (i.e., reports with severity *enhancement*) [8] are not included in the dataset.

Table 4
Features considered in this study and results of Wilcoxon-Mann-Whitney tests. Fig. 1 provides a visual representation of the features in the dimension of bug fix. A positive *d-value* indicates that non-negative bug reopens have significantly higher value on the feature, and a negative *d-value* indicates the opposite trend. The characters in parentheses represent the magnitude of the effect; N = negligible, S = small, M = medium, and L = large.

Dimension	Feature	CDT		JDT		PDE		Platform		Total	
		p-value	*d-value*	*p*-value	*d-value*	*p*-value	*d-value*	*p*-value	*d-value*	*p*-value	*d-value*
Work habit	Time of day	0.018	−0.357 (M)	0.821	0.013 (N)	0.653	0.065 (N)	0.033	−0.110 (N)	0.000	−0.135 (N)
	Day of week	0.940	−0.012 (N)	0.070	−0.102 (N)	0.772	−0.041 (N)	0.290	−0.054 (N)	0.062	−0.066 (N)
	Day of month	0.882	−0.023 (N)	0.575	0.032 (N)	0.559	−0.084 (N)	0.216	−0.064 (N)	0.280	−0.039 (N)
	Day of year	0.941	0.012 (N)	0.117	0.091 (N)	0.595	−0.077 (N)	0.928	−0.005 (N)	0.548	0.022 (N)
Bug report	Severity	0.376	0.102 (N)	0.202	0.059 (N)	0.557	0.053 (N)	0.363	−0.037 (N)	0.324	−0.027 (N)
	Priority	0.334	−0.033 (N)	0.279	−0.028 (N)	0.332	−0.030 (N)	0.028	−0.047 (N)	0.187	−0.017 (N)
Bug fix	Duration of survival (DS)	**0.000**	**−0.531 (L)**	**0.000**	**−0.338 (M)**	0.683	−0.060 (N)	**0.000**	**−0.236 (S)**	**0.000**	**−0.274 (S)**
	Duration of hidden (DH)	**0.001**	**−0.480 (L)**	**0.000**	**−0.211 (S)**	0.162	−0.201 (S)	**0.000**	**−0.276 (S)**	**0.000**	**−0.288 (S)**
	Duration of assignment (DA)	0.161	−0.067 (N)	0.000	−0.119 (N)	0.160	−0.060 (N)	0.000	−0.106 (N)	0.000	−0.104 (N)
	Duration of fixing (DF)	0.053	0.292 (S)	**0.000**	**−0.209 (S)**	0.270	−0.159 (S)	0.032	−0.111 (N)	0.004	−0.105 (N)
People	Number in CC list	0.397	−0.123 (N)	0.031	−0.122 (N)	0.352	−0.131 (N)	**0.000**	**−0.177 (S)**	**0.000**	**−0.162 (S)**
	Number of developers involved	**0.000**	**−0.602 (L)**	**0.000**	**−0.205 (S)**	**0.005**	**−0.384 (M)**	**0.000**	**−0.209 (S)**	**0.000**	**−0.214 (S)**

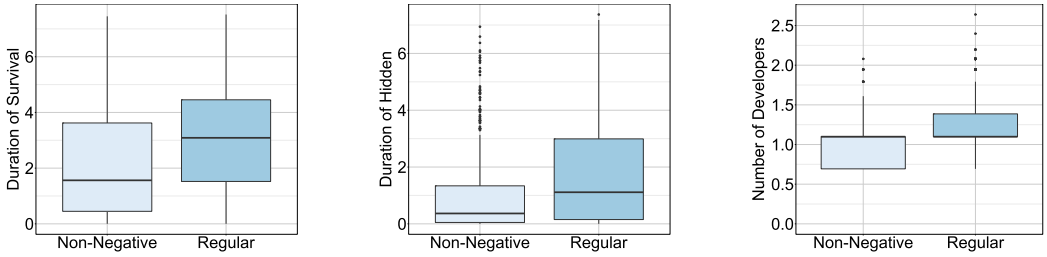

Fig. 2. Boxplots (in logarithmic scale) comparing non-negative and regular bug reopens.

5. Threats to validity

5.1. Scope of non-negative bug reopens

Initially, we intended to identify R4 (a better solution), R6 (misapprehension), and R7 (inappropriate status) within the range of non-negative bug reopens. However, after careful consideration, we restricted our study to the proportion of R6 and R7 that were categorized as either T3 or T4, for the following reasons.

R4 is an important category because bugs in R4 are reopened to be merged into modified code after being successfully fixed. Given that the main purpose of bug reopens in R4 is typically to improve program efficiency, we consider that they may affect end users significantly. Hence, R4 was excluded from the scope of this study.

To enhance the usability and efficiency of the identification method, we did not include the Eclipse CVS repository as a data source in this study. Thus, the proportion of R6 and R7 that were categorized into T1 or T2, respectively, were excluded.

Although a strict criterion was implemented to avoid false-positive and false-negative results, we do not claim that the proposed patterns are without limitations. Further research is required to improve the existing procedure; for instance, by combining it with other techniques or data sources.

5.2. Wrongly identified cases

During the examination process (Section 3.2), we have validated the proposed patterns with a precision of 92%, yet there are still several wrongly identified cases. For example, bug #292126 in the Platform project.

The assigned developer initially closed the issue with an INVALID label and requested further information:

- "Please reopen when you can get the stack traces."

The reporter reopened the bug to submit a thread dump. Because the information remained insufficient, the bug was closed with the same resolution:

- "We might need maybe another 2 or 3 thread dumps from while it's taking a long time."
- "Please reopen when you have more information."

The bug reopen originates from P2, so it would typically be classified as either R6 or R7. However, it actually belongs to R1, which indicates that our existing approach cannot reliably separate this type of bug reopen from others. Additional information is required to improve the identification procedure, such as the number of attachments (e.g., thread dumps) involved during the entire reopening process.

6. Conclusions and future work

Bug reopens are usually considered to be negative. However, the results of this study suggest that this is not always the case. Specifically, we made the following contributions:

- A novel concept of non-negative bug reopens was proposed. Moreover, we presented the reopen cycle and its corresponding taxonomy to enable the identification of non-negative bug reopens.
- A case study on Eclipse bug reports was examined. The results show that non-negative bug reopens are statistically significantly different from regular bug reopens, with at least a small effect size on the survival time (in CDT, JDT, and Platform) and the number of developers involved (in all four projects examined).

There are many possible directions for future research. For example,

it would be valuable to determine to what extent the presence or absence of non-negative bug reopens in training and test sets affects the effectiveness (accuracy) and efficiency (time) of prediction models. In addition, our empirical findings suggest a significant need for more targeted training for practitioners. This may reduce unnecessary reopens caused by limited technical knowledge and therefore minimize the corresponding maintenance costs as the ultimate goal.

Acknowledgments

We thank the anonymous reviewers for their constructive comments and suggestions. This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong (No. 11208017), and the research funds of City University of Hong Kong (No. 7004683).

References

[1] Q. Mi, J. Keung, An empirical analysis of reopened bugs based on open source projects, Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering - EASE '16, ACM Press, New York, New York, USA, 2016, pp. 1–10, http://dx.doi.org/10.1145/2915970.2915986.

[2] T. Zimmermann, N. Nagappan, P.J. Guo, B. Murphy, Characterizing and predicting which bugs get reopened, 2012 34th International Conference on Software Engineering (ICSE), IEEE, 2012, pp. 1074–1083, http://dx.doi.org/10.1109/ICSE.2012.6227112.

[3] E. Shihab, A. Ihara, Y. Kamei, W.M. Ibrahim, M. Ohira, B. Adams, A.E. Hassan, K.-i. Matsumoto, Studying re-opened bugs in open source software, Empirical Softw. Eng. 18 (5) (2013) 1005–1042, http://dx.doi.org/10.1007/s10664-012-9228-6.

[4] X. Xia, D. Lo, E. Shihab, X. Wang, B. Zhou, Automatic, high accuracy prediction of reopened bugs, Autom. Softw. Eng. 22 (1) (2014) 75–109, http://dx.doi.org/10.1007/s10515-014-0162-2.

[5] F. Thung, Lucia, D. Lo, L. Jiang, F. Rahman, P.T. Devanbu, To what extent could we detect field defects? an extended empirical study of false negatives in static bug-finding tools, Autom. Softw. Eng. 22 (4) (2015) 561–602, http://dx.doi.org/10.1007/s10515-014-0169-8.

[6] A. Jongyindee, M. Ohira, A. Ihara, K.-i. Matsumoto, Good or bad committers? — A case study of Committer's activities on the Eclipse's bug fixing process, IEICE Trans. Inf. Syst. E95.D (9) (2012) 2202–2210, http://dx.doi.org/10.1587/transinf.E95.D.2202.

[7] A. Lamkanfi, J. Perez, S. Demeyer, The Eclipse and Mozilla defect tracking dataset: A genuine dataset for mining bug information, 2013 10th Working Conference on Mining Software Repositories (MSR), IEEE, 2013, pp. 203–206, http://dx.doi.org/10.1109/MSR.2013.6624028.

[8] K. Herzig, S. Just, A. Zeller, It's not a bug, it's a feature: How misclassification impacts bug prediction, 2013 35th International Conference on Software Engineering (ICSE), IEEE, 2013, pp. 392–401, http://dx.doi.org/10.1109/ICSE.2013.6606585.

[9] Y. Tian, M. Nagappan, D. Lo, A.E. Hassan, What are the characteristics of high-rated apps? A case study on free Android Applications, 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, 2015, pp. 301–310, http://dx.doi.org/10.1109/ICSM.2015.7332476.

[10] B. Kitchenham, L. Madeyski, D. Budgen, J. Keung, P. Brereton, S. Charters, S. Gibbs, A. Pohthong, Robust statistical methods for empirical software engineering, Empirical Softw. Eng. 22 (2) (2016) 1–52, http://dx.doi.org/10.1007/s10664-016-9437-5.